Database Concepts Final Project

You have been hired by AdventureWorks Bikes Retailer (AW Bikes) as a Database Analyst. Your first project includes making updates to their sales database and writing a number of reports.

**Initial Steps:**

* ~~Examine the structure of the database. Look at the data in each table. Understand what each column’s data represents.~~
* ~~Build an ERD of the database in its initial state. This should include each entity in the database with all of the columns & datatypes. You will need to identify the primary and foreign keys, as well as the relationships between the entities.~~

**Database Updates:**

1. Management would like the ability to track additional information about product categories. They would like a new ProductCategory table to be created for this purpose. The current product category should get moved to the new table. You will need to add a column to the Product table for the new ProductCategory ID, populate it with the correct ID, and then remove the existing ProductCategory Column.

Create table ProductCategory(

CategoryID int identity(1,1),

CategoryName varchar(50)

)

Insert into ProductCategory(CategoryName)

Select distinct Category

from Product

Alter table product

add CategoryID int

update Product

set CategoryID = c.CategoryID

from Product p

join ProductCategory c

on p.Category = c.CategoryName

Alter table Product

drop column Category

1. Management would also like one table that stores address information, instead of having the customer’s home address in the Customer table, and their shipping address on the OrderHeader table. Create a new table for addresses. This should store the customer’s ID, the address type (Shipping or Home), and the address fields. Populate the new addresses into the new address table, then update the Customer and OrderHeader tables to reflect the correct addressID. Finally, remove the old address fields from Customer and OrderHeader.

Create table CustomerAddresses(

AddressID int identity(1,1),

CustomerID int,

AddressType varchar(10),

Address varchar(50),

City varchar(30),

State char(2),

ZipCode varchar(10)

)

Insert into CustomerAddresses(CustomerID, AddressType, Address, City, State, ZipCode)

Select CustomerID, 'Home', Address, City, State, ZipCode

from Customer

Insert into CustomerAddresses(CustomerID, AddressType, Address, City, State, ZipCode)

Select CustomerID, 'Ship', ShipAddress, ShipCity, ShipState, ShipZipCode

from OrderHeader

alter table Customer

Add AddressID int

alter table OrderHeader

add AddressID int

update Customer

set AddressID = a.AddressID

from Customer c

join CustomerAddresses a

on c.CustomerID = a.CustomerID

where a.AddressType = 'Home'

update OrderHeader

set AddressID = a.AddressID

from OrderHeader o

join CustomerAddresses a

on o.CustomerID = a.CustomerID

where a.AddressType = 'Shipping'

alter table Customer

drop column Address

alter table Customer

drop column City

alter table Customer

drop column State

alter table Customer

drop column ZipCode

alter table OrderHeader

drop column ShipAddress

alter table OrderHeader

drop column ShipCity

alter table OrderHeader

drop column ShipState

alter table OrderHeader

drop column ShipZipCode

1. Management would also like you to add foreign keys to each table in the database where they make sense.

alter table CustomerAddresses

add constraint pk\_CustomerAddresses\_AddressID Primary Key (AddressID)

alter table Customer

add constraint fk\_Customer\_AddressID Foreign Key (AddressID)

references CustomerAddresses(AddressID)

alter table OrderHeader

add constraint fk\_OrderHeader\_CustomerID Foreign Key (CustomerID)

references Customer(CustomerID),

constraint fk\_OrderHeader\_AddressID foreign key (AddressID)

references CustomerAddresses(AddressID)

alter table CustomerAddresses

add constraint fk\_CustomerAddresses\_CustomerID Foreign Key (CustomerID)

references Customer(CustomerID)

alter table ProductCategory

add constraint pk\_ProductCategory\_CategoryID primary key (CategoryID)

alter table Product

add constraint fk\_Product\_VendorID Foreign Key (VendorID)

references Vendor(VendorID),

constraint fk\_Product\_CategoryID foreign key (CategoryID)

references ProductCategory(CategoryID)

alter table OrderDetail

alter column SalesPromotionID smallint

alter table OrderDetail

add constraint fk\_OrderDetail\_OrderID foreign key (OrderID)

references OrderHeader(OrderID),

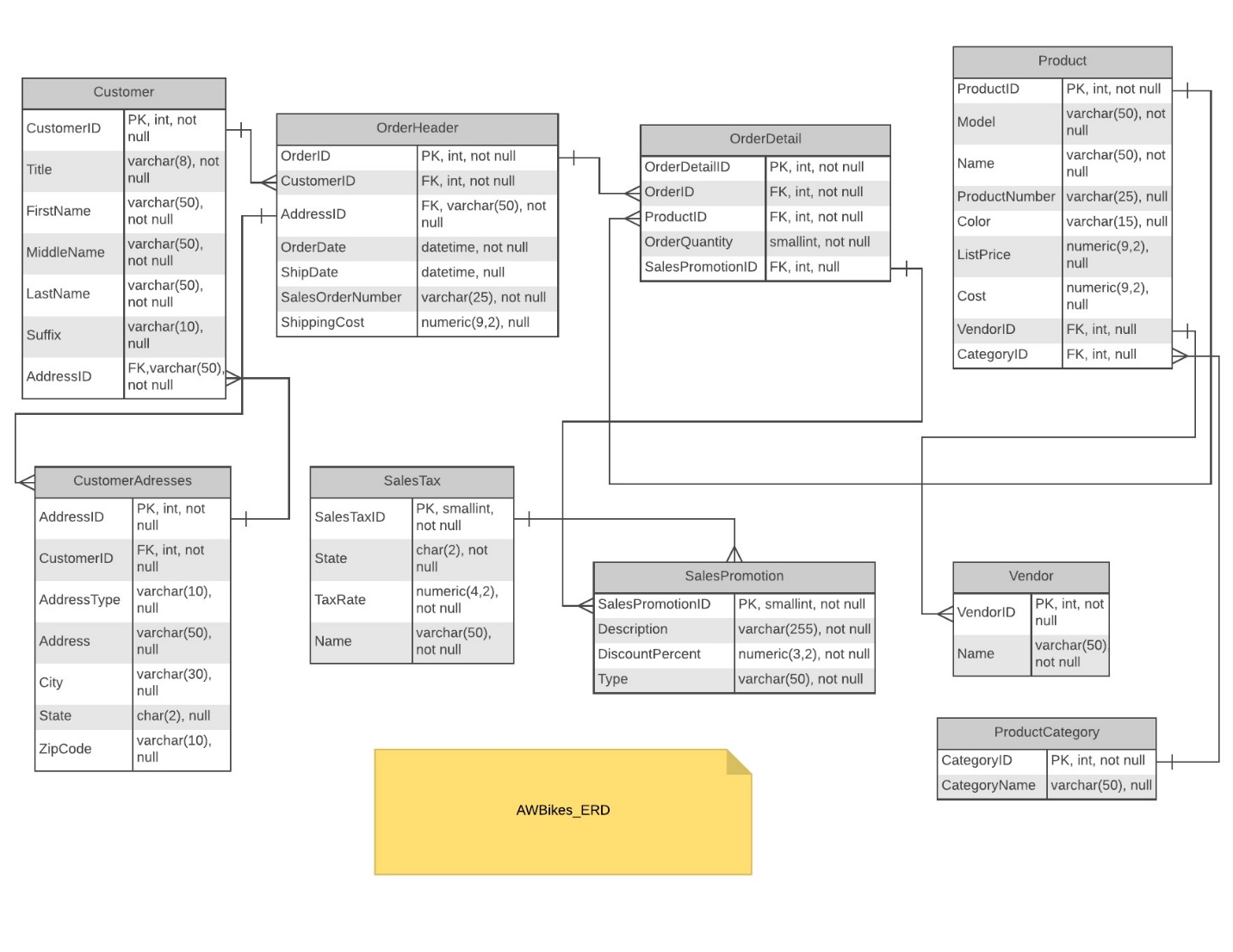
constraint fk\_OrderDetail\_ProductID foreign key (ProductID)

references Product(ProductID),

constraint fk\_OrderDetail\_SalesPromotionID foreign key (SalesPromotionID)

references SalesPromotion(SalesPromotionID)

1. After making these changes to the database, update your ERD to reflect the new database structure.



**Reports:**

Management has defined three reports that the need to be written. AW Bikes uses Microsoft Access as their reporting software. Management requires that all reports written use a view as their data source, in order to decrease the effort needed to modify the reports if the source of the data needs changes.

1. Product Report:

Management needs a report of all of their products listed by product category. This report should list all information about the product, including the profit generated, and the name of the vendor.

Create View ProductInformationByCategory\_VW as

Select p.ProductID, p.Name as ProductName, p.Model, p.ProductNumber, p.Color, p.ListPrice, p.Cost, (p.ListPrice - p.Cost) as Profit, v.VendorID, v.Name as VendorName, pc.CategoryID, pc.CategoryName

from Product p

left join ProductCategory pc

on p.CategoryID = pc.CategoryID

left join Vendor v

on p.VendorID = v.VendorID

Select \*

from ProductInformationByCategory\_VW

1. Sales Report By Vendor:

Management needs a report of sales by vendor. This report should list totals of all sales by vendor for a 2014. Total sales should reflect the quantity of an item a customer ordered. Don’t forget to apply the discount. Include the vendor name, the total sales generated by products from the vendor, the total discounts for each vendor, and the total profit generated by the vendor. Only one record should be listed for each vendor. The report should also show grand totals at the bottom.

create view SalesReportByVendor\_VW as

Select v.VendorID, v.Name as VendorName, cast(sum(p.ListPrice \* (1 - sp.DiscountPercent) \* od.OrderQuantity) as decimal(15,2))TotalSales, sum(od.OrderQuantity) as TotalQuantity, cast(sum(p.ListPrice \* sp.DiscountPercent \* od.OrderQuantity)as decimal(8,2)) as TotalDiscount, cast(sum((p.ListPrice \* (1-sp.DiscountPercent)) - p.Cost)as decimal(15,2)) as TotalProfit

from OrderHeader oh join OrderDetail od

on oh.OrderID = od.OrderID

join SalesPromotion sp on od.SalesPromotionID = sp.SalesPromotionID

join Product p on od.ProductID = p.ProductID

join Vendor v on p.VendorID = v.VendorID

where oh.OrderDate between '1/1/2014' and '07/31/2014'

group by v.VendorID, v.Name

Select \*

from SalesReportByVendor\_VW

1. Order Invoice:

With the addition of the Address table, Management needs a new Order Invoice report. The output of the report will be included in the customer’s package. This report should be able to be run for a particular Sales Order Number (You will need to include a prompt/parameter to do this). The order invoice should include the Order Date, Sales Order Number, customer name, home address, and shipping address; each product ordered, the retail price of the product, the discount amount, and the line total; the order subtotal, tax, and total.

Create view OrderInvoice as

Select o.OrderDate,

o.SalesOrderNumber,

c.FirstName + ' ' + c.LastName as CustomerName,

Isnull(ca1.Address, '') + ' ' + Isnull(ca1.City, '') + ' ' + Isnull(ca1.State, '') + ' ' + Isnull(ca1.ZipCode, '') As HomeAddress,

Isnull(ca2.Address, '') + ' ' + Isnull(ca2.City, '') + ' ' + Isnull(ca2.State, '') + ' ' + Isnull(ca2.ZipCode, '') As ShippingAddress,

p.Name as ProductName,

p.ListPrice as RetailPrice,

Cast((p.ListPrice \* sp.DiscountPercent) as Decimal(8,2)) as DiscountAmount,

od.OrderQuantity,

Cast(p.ListPrice \* (1-sp.DiscountPercent) \* od.OrderQuantity as Decimal(10,2)) as LineTotal,

isnull(st.TaxRate,0) as TaxRate,

Cast((p.ListPrice \* (1-sp.DiscountPercent) \* od.OrderQuantity) \* (isnull(st.TaxRate,0)/100) as Decimal(8,2)) as SalesTax

From

OrderHeader o

Join Customer c on o.CustomerID = c.CustomerID

Join CustomerAddresses ca1 on c.AddressID = ca1.AddressID And ca1.AddressType = 'Home'

Join CustomerAddresses ca2 on o.AddressID = ca2.AddressID And ca2.AddressType = 'Shipping'

Join OrderDetail od on o.OrderID = od.OrderID

join Product p on od.ProductID = p.ProductID

Left Join SalesPromotion sp on od.SalesPromotionID = sp.SalesPromotionID

Left Join SalesTax st on st.State = ca1.State

select\*

from OrderInvoice

where SalesOrderNumber='SO58366'

**Niceties:** [Extra Credit]

1. Management would like you to create a function that handles the sales tax calculation. They feel it is important to have a consistent way to calculate the sales tax, and feel that creating a function will accomplish this.
2. AW Bikes will rely on the reports you have written and they will be run often. Order data will continue to grow. Management would like you to analyze the queries that run the reports and add indexes that will keep them running quickly.

Deadlines and Additional Information

**Deadlines:**

The initial version of the ERD will be due Tuesday, Nov 15th at 12:30 PM. Submit on Blackboard in the dropbox located in the Project folder. **Late submissions will not be accepted.**

The remainder of the project is due Tuesday, December 20th at 2:30PM. **No exceptions will be made and no work will be accepted after this time.**

**Submitting the Final Project:**

ERDs should be submitted as PDFs. No other formats will be accepted. The final project should be submitted as a zip folder containing the following:

1. Final ERD
2. A document with the SQL commands used for items 1-3 under the Database Updates section, the definitions of the views/stored procedures used for the basis of the reports, and the SQL commands for any of the extra credit
3. PDF of report output and query for each report.

**Point Distribution:**

|  |  |
| --- | --- |
| **Required Components** |  |
| Initial ER Diagram | 20 |
| Modifications to ER Diagram | 5 |
| Table Changes | 30 |
| Foreign Keys | 5 |
| Product Report | 5 |
| Sales By Vendor | 10 |
| Invoice | 25 |
| **Total** | 100 |
| **Extra Credit** |  |
| Function to calculate order sales tax | 5 |
| Indexes to support report queries | 5 |

**ERD:**

The ERD should be created in some sort of software for this purpose (Visio, LucidChart, Gliffy, etc). All tables and columns should be included. The relationships between the entities should be visible, and should denote the type of relationship (1:1, 1:M, M:1).

**Reports:**

There will be three components to the grading of the reports.

|  |  |
| --- | --- |
| **Component** | **%** |
| Accurate Data | 50 |
| SQL Syntax - Best Practices, Use of View | 10 |
| Design/Layout of Report | 40 |

Accurate Data:

For full points in this category, the data must be 100% accurate. All calculations must be correct. All records that should appear do, and there are no records that shouldn’t appear.

SQL Syntax:

For full points, your report queries must include the correct usage of joins, where/having clause conditions, and the use of a view as the source of the data.

Design/Layout of Report:

For full points, the report should include a title that is descriptive of what the report is (IE, “Product List”, NOT “Report 1”), and a report footer with page number. Information should be displayed clearly, in a logical manner, and should be easy to find. Columns should have headings where they make sense, appropriate labels should be used, and the data should be aligned. The report should look professional, like something that a company would actually use.

**Extra Credit:**

1. A scalar function should be created that calculates the sales tax of an order. Method is up to the student. The function must correctly calculate the sales tax based on the subtotal and the state the order was placed in.
2. Analyze the queries created for the reports. Create indexes where they will assist in pulling the information quickly. Points on this will be determined by the ability to identify where indexes are needed, as well as the appropriateness of the indexes created.